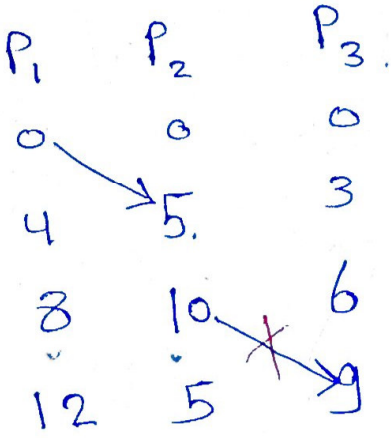
slide 4
synchronization in Distributed system

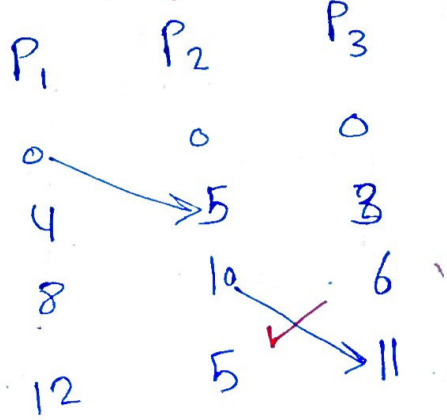
LamPort Logic clock:

when a machine send Message at t_0 to another machine at t_1 so t_1 must be greater than t_0 Logically.

wrong



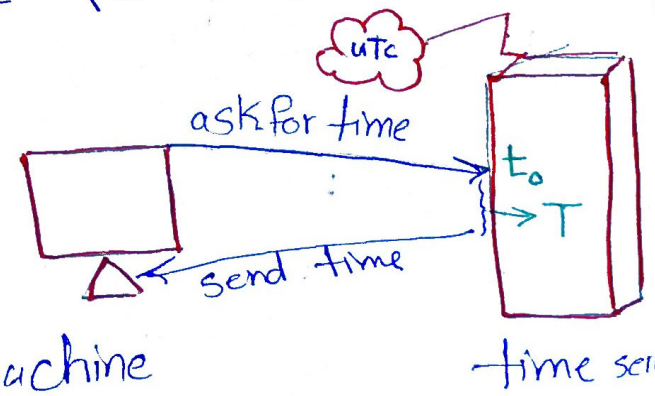
right



- two ways to maintain synchronization :

cristian's Algorithm (using uTc)

- You have a time server each machine ask him for time and he references uTc (universal cordinated time).



Assume :

RTT = ask time + send time

&

ask time = send time

So Send time = RTT/2

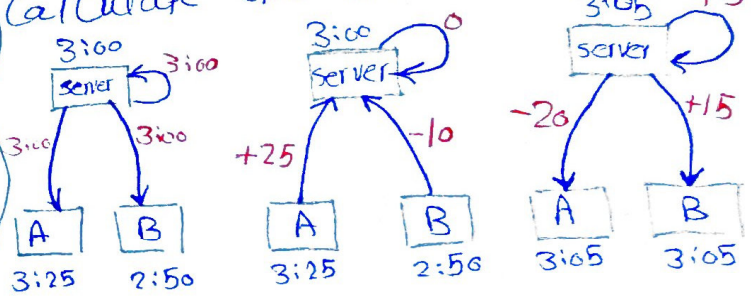
t_0 : request Arrive time

Given time to Machine = $t_0 + T + \frac{RTT}{2}$

Berkeley unix (without uTc)

1- centralized :

time server send its time to all machines and they respond by difference between them then he sends correcting time to each one after calculate the difference average



2- Decentralized : all Machines broadcast its time and calculate the average and correct only its own time

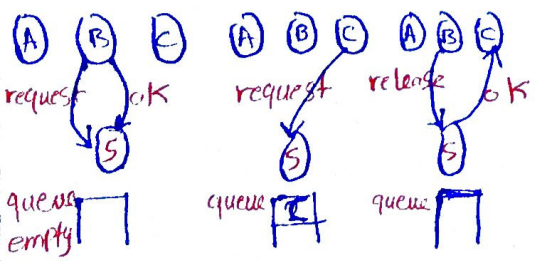
- two APProaches to Maintain mutual exclusion of shared memory :

[1] Permission [2] Token

Permission

Centralized

one Machine is eleded as a Cordinator and any Machine want to access any resource asking him and he reply with ok if the resource is free and no reply and Make a machine in ~~res~~ the resource queue if it's used by another one, until it's released, then give him Permission.



advantage : • simple to implement

disadvantages : • single Point of failure
• bad Performance

Decentralized

• each resouce is replicated n times ~~and~~ with one Cordinator per replica.

• when a machine want to access some resources search by resource name in hash table when Machine have it and to have Permission to Access it Must get Permission from ~~machine~~ resource

Cordinator > n/2

advantage : No single Point of failure.

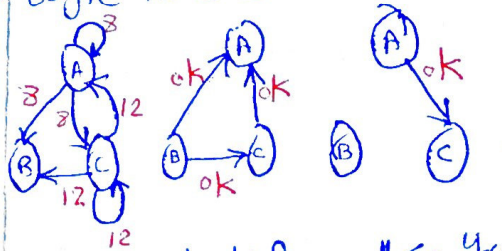
disadvantage : lots of Messages.

Distributed

• resource requestor send Messages to all Machines with its Logic time and which resouce he want.

if it's busy, he will get into queue.

if two Machine ask at the same time, the least Logic time will win.



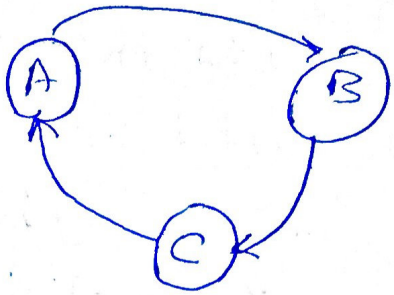
if you get ok from all so you win.

advantage : • No centralization
• fewer Messages

disadvantage : n Point of failure as you have get ok Message from all.

[2] Token system

organize the processes (Machines) in a ring and Path a token through the ring, the one has the token can access any resources if it doesn't need it Path it to the successor



advantage:

- fairness (No starvation)

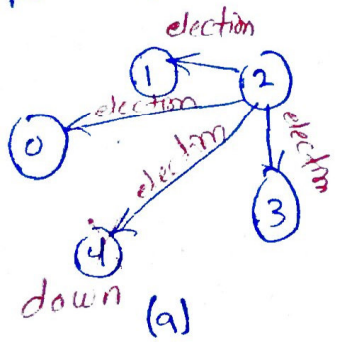
disadvantage:

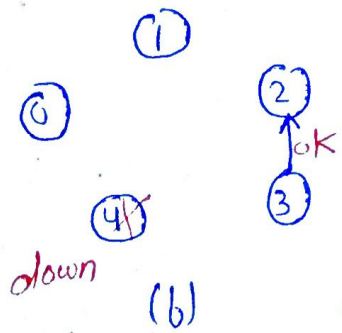
- crash of a process holding token
- if token is lost, it's to be regenerated

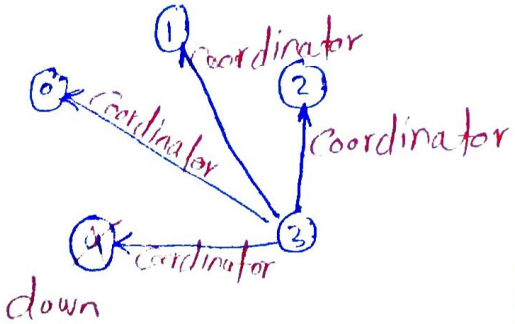
Election algorithms for Coordinator:

① Bully Algorithm

each process has a number, each one would like to be coordinator send message to upper-numbered processes and if any of them inside the election she send ok-message to it to tell it, he's alive and stop him until the biggest number when

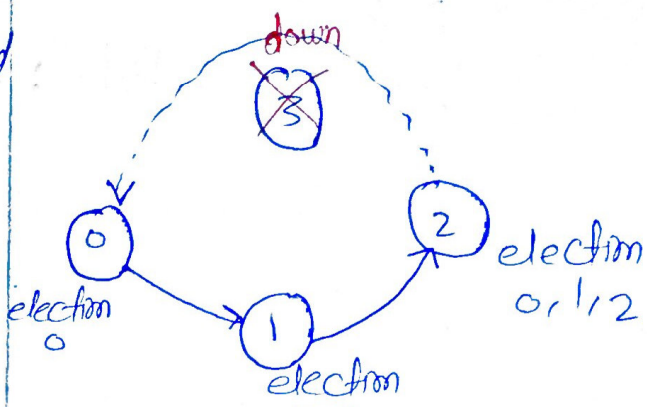


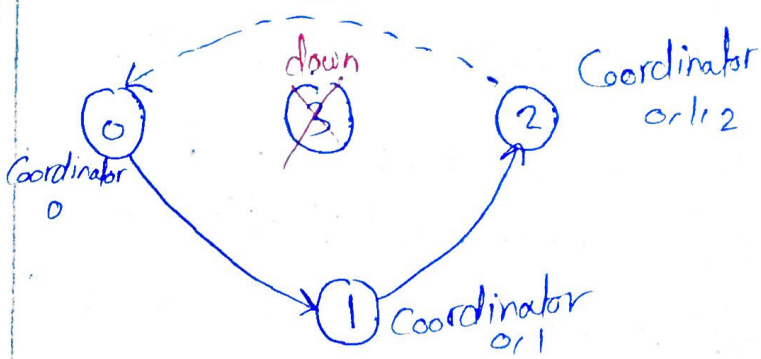


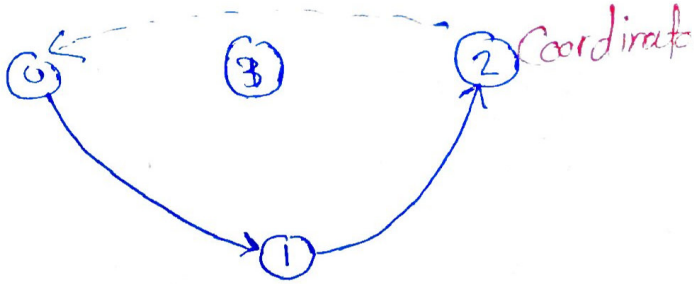


② Ring Algorithm

each process also has a number, and each one would like to be Coordinator send Message and one with highest number, win the election.







Transaction : one or more operations happens Completely (commit) or not at all (Rollback) (ex: transfer money between banks accounts)

types of transcations:

1- flat transaction (processing on one database).

2- Nest transcation (processing on two different DB).

airLine DB — Hotel DB

3- Distributed transcation (processing on the same DB but seperated physically)

Methods of implementing transcation:

[1] private workspace

You make changes on a copy of real data in your workspace after complete it, reflect them on real data. and if it's not completed, No effect on real data

disadvantage: cost of copy files to workspace.

[2] write ahead Log

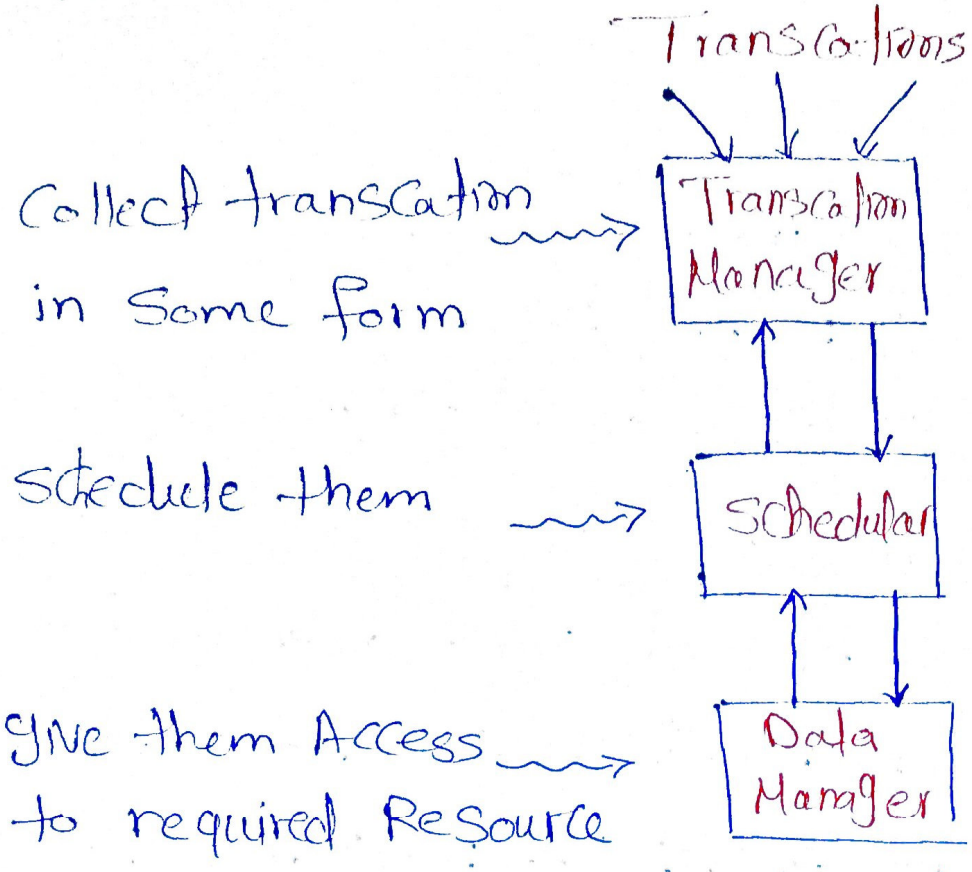
You make changes on the real data but you log its previous value before change, if it's aborted (not complete transcation) get back old value.

Two Phase Commit Protocol : most widely protocol used for implement transcation in Distributed system.

phase 1 : The Coordinator send Prepare message to all Machine to check if they are ready to implement transcation, if he get ready Messages from all of them, he send Commit order to them

phase 2 : after they finish, they Notifie him. if he doesn't get ready message from all, the transcation is aborted (Cancelled)

handling Mutual exclusion (concurrency control) between transactions:



three Methods of Scheduling (concurrency control)

Lock / Release

when a process access any resource, it locks it ~~on~~ till it finishes the task, then release if, so I ensure from Mutual exclusion.

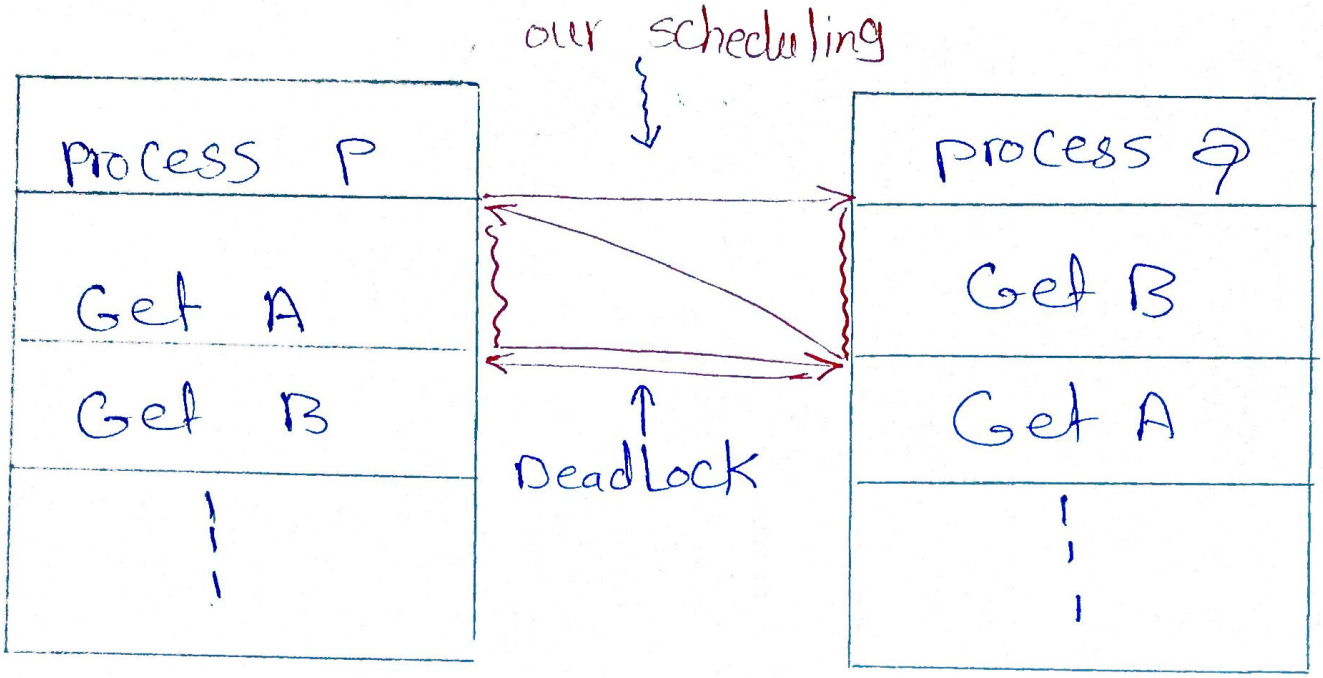
optimistic

assume every process x no others works on the resources and if the resources Status changed during transaction by others, the transaction is aborted, otherwise it's committed.

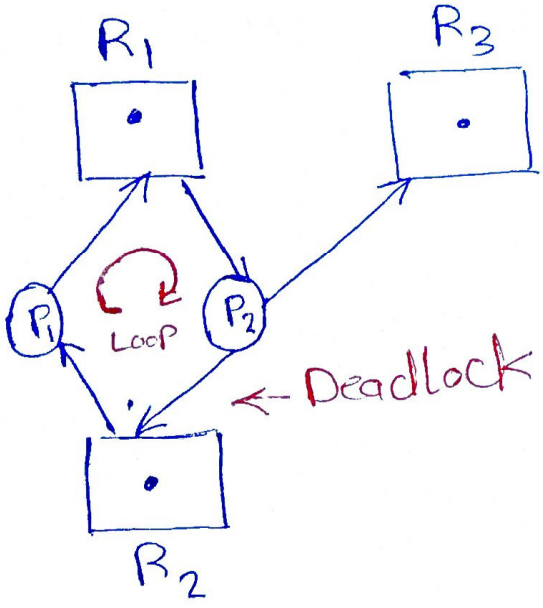
Time Stamp

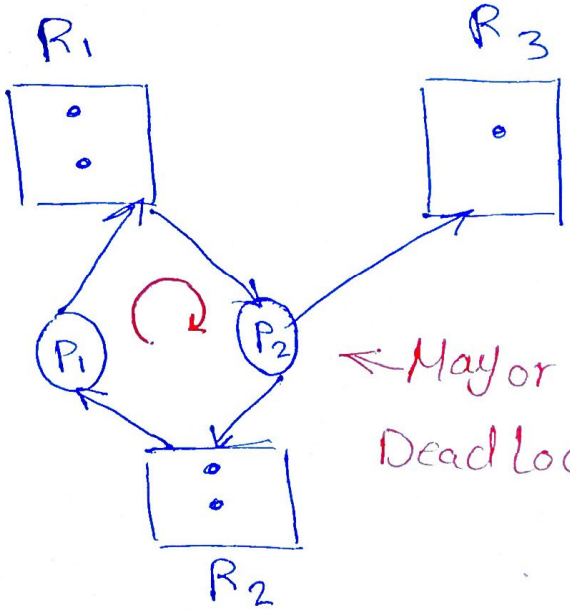
every process give the accessed resource its time stamp when Accessing it, and if the resource has higher timestamp at any moment, meaning another one Access it so it's aborted, otherwise it's committed.

the Deadlock problem : a process hold the Resource A and wait for the resource B which is hold by another process waiting Resource A. :)

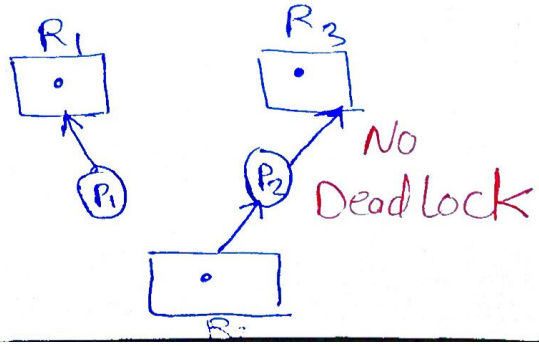


Resource Allocation Graph





: Mean Resource
: Process
: n. of Replication
: Request resource
: resource held by this process



Dead Lock Characterization

" not understanding yet "

slide number (83)